WCCI 2012 IEEE World Congress on Computational Intelligence
June, 10-15, 2012 - Brisbane, Australia IEEE CEC

Comparative study of visualisation methods for temporal data

Tzai Der Wang
Department of Industrial Engineering and Management,
Cheng Shiu University, Taiwan, ROC
tw952276@gmail.com
Xiaochuan Wu and Colin Fyfe
University of the West of Scotland, Scotland
Xiaochuan.Wu,colin.fyfe@uws.ac.uk

***Abstract*—In this paper, we investigate a financial data set from [5] using two algorithms which are both designed for visualising data. One algorithm consists of a neuroscale algorithm which uses different Bregman divergences. The other uses a similar algorithm but based on reservoir computing. We show that the latter is much better because it captures the dynamical nature of the financial time series and thus reveals more explicit information. By investigating a slightly different cost function, we show that the latter mapping is not creating information which does not exist in the original time series.**

I. Introduction

Exploratory data analysis forms a group of methods which are designed to interrogate data in order to identify some previously unknown aspects of the data: typically the original data will be high dimensional so the aspects uncovered will not be easily seen by eye. One possibility is to project the data onto a low dimensional manifold and investigate, by eye, what structure can be seen in the projection. The implicit assumption in this type of method is that the structure seen in the projection is present in the original high dimensional data.

One group of methods which tries to ensure that this implicit assumption is met is known as multidimensional scaling. One disadvantage of multidimensional scaling is that it creates one point on the projection manifold for each data point and every time a new data point is added the manifold may change. Neuroscale [8] is a visualisation algorithm designed to overcome this shortcoming.

We have recently created algorithms which extend the neuroscale algorithms in two ways:

1) By using Bregman divergences instead of Euclidean distances [10].
2) By using reservoirs to capture the dynamic information in a data set [12].

In this paper, we compare these two methods in the context of a particular financial data set.

II. Background

In this section, we review the three technologies on which the algorithms in this paper will be based: multidimensional scaling including neuroscale, Bregman divergences and reservoir computing.

A. Multidimensional Scaling and Neuroscale

Metric multidimensional scaling is a group of methods designed to extract information from a high dimensional data set by projecting the data set onto a lower dimensional, generally 2 dimensional, manifold in which the structure of the original high dimensional data can be discerned by eye. This task is performed by associating with each high dimensional data sample, a low dimensional latent space point. The latent space points are moved around so as to minimise the sum of the differences between the latent space distances and the data space distances.

More formally, let the data samples be $\mathbf{x}_i, i = 1, ..., N$ and let the corresponding latent points be $\mathbf{y}_i, i = 1, ..., N$. Let D_{ij} be the (Euclidean) distance in data space between points $\mathbf{x}_i$ and $\mathbf{x}_j$ and let L_{ij} be the corresponding distance between latent points $\mathbf{y}_i$ and $\mathbf{y}_j$. Then in classic MDS, the latent points are moved to minimise

$$E_{MDS} = \sum_{i,j} (D_{ij} - L_{ij})^2 \tag{1}$$

which is known as the stress function of MDS.

There are a great number of variants upon this, the most popular being the Sammon mapping whose stress function is

$$E_{Sammon} = \frac{1}{Z} \sum_{i,j} \frac{(D_{ij} - L_{ij})^2}{D_{ij}} \tag{2}$$

where Z is a normalising factor.

The neuroscale mapping [8] is based on a radial basis network: let the input vector be $\mathbf{x}$; then the response of the i^{th} hidden neuron is given by $h_i = \exp(-\frac{\|\mathbf{x}-\mathbf{c}_i\|}{\sigma})$ and the output, y_j is given by $y_j = \sum_i w_{ji} h_i$. Typically only the w_{ij} parameters are changed during learning which is generally supervised so that each input $\mathbf{x}$ must be associated with a target value, $\mathbf{t}$. The neuroscale algorithm replaces this supervised learning with a new training process known as 'relative supervision': instead of error descent with respect to the target values, neuroscale uses the classical MDS stress function (1) as the error to be minimised.

B. Bregman divergences

Consider a strictly convex function $F : S \to \Re$ defined on a convex set $S \subset \Re^d$ ($\Re^d$ denotes d-dimensional real vector space); a Bregman divergence [9], [6] between two points, $\mathbf{p}$ and $\mathbf{q} \in S$, is defined to be

$$d_F(\mathbf{p}, \mathbf{q}) = F(\mathbf{p}) - F(\mathbf{q}) - \langle (\mathbf{p} - \mathbf{q}), \nabla F(\mathbf{q}) \rangle, \qquad (3)$$

where the angled brackets indicate an inner product and $\nabla F(\mathbf{q})$ is the derivative of F evaluated at $\mathbf{q}$. It can be thought of as the difference between $F(\mathbf{p})$ and the first order Taylor series expansion of $F(\mathbf{p})$ around $F(\mathbf{q})$ [6].

In the following we use the Itakura-Saito (IS) divergence based on $F(x) = -\log x$ which gives

$$d_F(L_{ij}, D_{ij}) = -\log L_{ij} + \log D_{ij} - (L_{ij} - D_{ij}) \left(\frac{-1}{D_{ij}} \right) \quad (4)$$

We call this the IS Left divergence since the distance we can change by moving the latent positions is in the left position. Similarly $d_F(D_{ij}, L_{ij})$ is called the IS Right divergence.

The Generalised Information (GI) divergence is based on $F(x) = x \log x$ and we also refer to one member of a family of divergences parameterised by t discussed in [7]. This is based on the function $F(x) = \frac{1}{x}$ with $t = 3$.

C. Reservoir Computing

There are two main types of reservoirs:

- The liquid state machine [11]
- Echo state networks [1], [3]

We will concentrate on the latter.

Echo state networks (ESNs) consist of three layers of 'neurons': an input layer which is connected with random and fixed weights to the next layer which forms the reservoir. The neurons of the reservoir are connected to other neurons in the reservoir with a fixed, random, sparse matrix of weights. Typically only about 10% of the weights in the reservoir are non-zero. The reservoir is connected to the output neurons using weights which are trained using error descent. We emphasise that only the reservoir to output weights are trainable; the other sets of weights are fixed. It is this feature which gives the ESN the property of being easily and efficiently trained.

We first formalise the idea of reservoir. W_{in} denotes the weights from the N_u inputs $\mathbf{u}$ to the N_x reservoir units $\mathbf{x}$, W denotes the $N_x \times N_x$ reservoir weight matrix, and W_{out} denotes the $(N_x + 1) \times N_y$ weight matrix linking the reservoir units to the output units, denoted $\mathbf{y}$. Typically $N_x \gg N_u$. W_{in} is fully connected and fixed (i.e. the weights are non-trainable). Similarly W is fixed but provides sparse connections: in this work only 10% of the weights in W are non-zero. W_{out} is a fully connected set of trainable weights (the "readout weights"). The W weights are such that the spectral radius (its greatest eigenvalue) is less than 1 which ensures stability in the reservoir. The larger it is the more memory of previous values can be retained.

The network's dynamics are governed by

$$\mathbf{x}(t) = f(W_{in}\mathbf{u}(t) + W\mathbf{x}(t-1)) \qquad (5)$$

where typically $f(.) = \tanh(.)$ and t is the time index. The feed forward stage is given by

$$\mathbf{y} = W_{out}\mathbf{x} \qquad (6)$$

The weights, W_{out}, are typically updated using error descent. We have previously [12] used the multidimensional scaling criterion in a manner similar to the neuroscale algorithm.

III. Simulations

A. Bregmanised Neuroscale

As in [12], we use a set of financial data from [5]: this data set is composed of the closing price at the start of each month of the S & P Composite market. We have used this closing price, the dividend, earnings, consumer price index and the long term interest rate to form a 5 dimensional data set. We have taken the first 1000 samples as the training data.

In this paper, we set the size of the reservoir, N_x, to 300, and investigate different

- structures of reservoirs: we have used
  - standard random reservoirs with approximately 10% non-zero weights,
  - a simple cyclic reservoir [4] in which reservoir neuron n is connected to neuron $n + 1$ only and neuron N_x is connected to neuron 1
  - Scale Free. The scale free property [2] is such that the probability that a neuron is connected to k other neurons follows a power law distribution, $P(k) \propto k^{-\alpha}$. Again the magnitude of the non-zero weights is random, typically drawn from a uniform distribution in [-1,1]. This gives a network in which most neurons are connected to very few others while a few neurons have many non-zero connections. The scale free property is said to be exhibited by academic collaborators and the world wide web.
  - Small world. This type of network is defined in terms of its average path length between neurons and gives rise to a clustering coefficient. This type of network is closely linked to the scale free network and again it has been suggested that academic collaborators form a small world network as well as showing the famous "six degrees of separation".
- Bregman divergences including the IS, Generalised Information, and two families of algorithms from [7]. Note that we are calculating the Bregman divergences between D_{ij} and L_{ij}: we are not using the Bregman divergences to calculate L_{ij}.

We show in Figure 1 the results when using the standard neuroscale algorithm on this data. The top figure shows the projection of the whole training data (1000 samples, each month beginning in 1871); we can see structure in that there is a continuous set of projections. The second figure shows every 12th sample; this gives the projection of each January sample every year from 1871 onwards with the numbers indicating the number of years after 1871. (Thus sample 35 represents January 1906). Some structure is obvious: we see the 1940s and 50s at the left of the diagram and the 1870s on the right of the diagram but many features are very obscure in this diagram.

Figure 2 shows the corresponding results when we use the Itakura Saito divergence, with the projections in the left position. We see a slightly better projection in that some of the

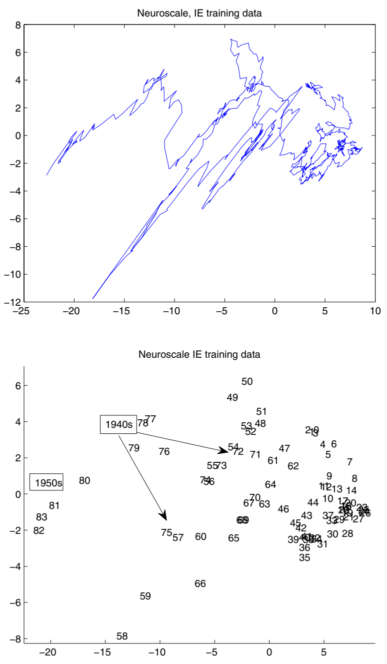

Fig. 1. Projection of the "Irrational Exuberance" data [5] with the standard neuroscale algorithm. Top: 1000 training samples. Bottom: projection of January of each year as an offset from 1871.

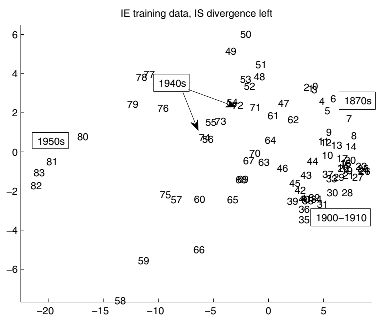

Fig. 2. Projection of the "Irrational Exuberance" training data [5] with the Itakura Saito Left divergence.

intervening years are more clearly definable but nevertheless there are still areas where there is little in the way of informative projections.

Similarly, Figure 3 shows the results when we use the GI divergence with the projection in the left position. Again we can see that some features are perhaps a little clearer than those from the original neuroscale algorithm. We have similar results with other divergences and so our overall conclusion is that the Bregman divergences may assist in uncovering information from this financial time series but with only limited success.

B. *Using reservoirs*

There now exist two possibilities for calculating D_{ij}:

1) $D_{ij} = \parallel \mathbf{u}_i - \mathbf{u}_j \parallel$. This is in line with the original neuroscale algorithm [8] and simply substitutes the non-linearity of the radial basis functions with that of the reservoir. This method has the advantage that the D_{ij} values need only be calculated once before training commences since they will not subsequently change.
2) $D_{ij} = \parallel \mathbf{x}_i - \mathbf{x}_j \parallel$. This has the advantage that D_{ij} then takes into account the history of the time series which is echoing in the reservoir. This is something that the first formulation would not allow.

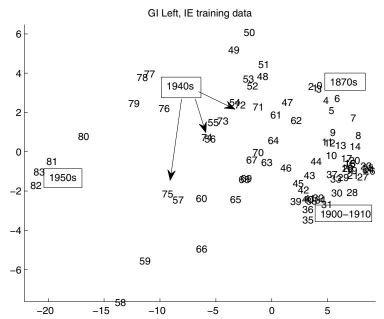

Fig. 3. Projection of the "Irrational Exuberance" training data [5] with the Generalised Information Left divergence.

We have previously only considered the second distance and this is the option that we use in this section, with the first option being considered in the next section. Note again that we are calculating the Bregman divergences between the Euclidean distances D_{ij} and L_{ij}.

Figure 4 shows exemplar results when we use the Euclidean distance; the top 2 diagrams use a random reservoir, while the bottom 2 use a small world reservoir. Perhaps the most striking thing about these projections is that the data set is clearly forming two distinct clusters. Close examination shows that the year 1918 formed the break between one cluster and the next. The first cluster contains the early years of the data set and shows little structure of interest: there are local areas where, for example, the 1870s predominate but little pattern to the movement of the projection between the years. We can also see that during the 1920s and 30s there was a greater spread than at other times while there is a cyclical element to the mapping during the later decades.

This cyclical element is also apparent in Figure 5 which gives the same projection using a random reservoir and the IS right divergence. We can also identify the transition year 1918 with the values from 1917 seeming to be striving to leave the first cluster.

Thus we see that we can gain much more information from the use of a reservoir.

C. *Using Data Space distances*

However the above method leaves open the possibility that we are creating structure in the projection when no such structure exists in the original data: using the distance in reservoir space to form the projection is explicitly taking into account at least some of the recent history of the time series.

Thus in this section, we investigate the other option - to use $D_{ij} = \parallel \mathbf{u}_i - \mathbf{u}_j \parallel$, the distance in data space, as the target for the distances in the final visualisation space.

We show the best results (over all reservoir structures and divergences) when using the data space distances in Figures

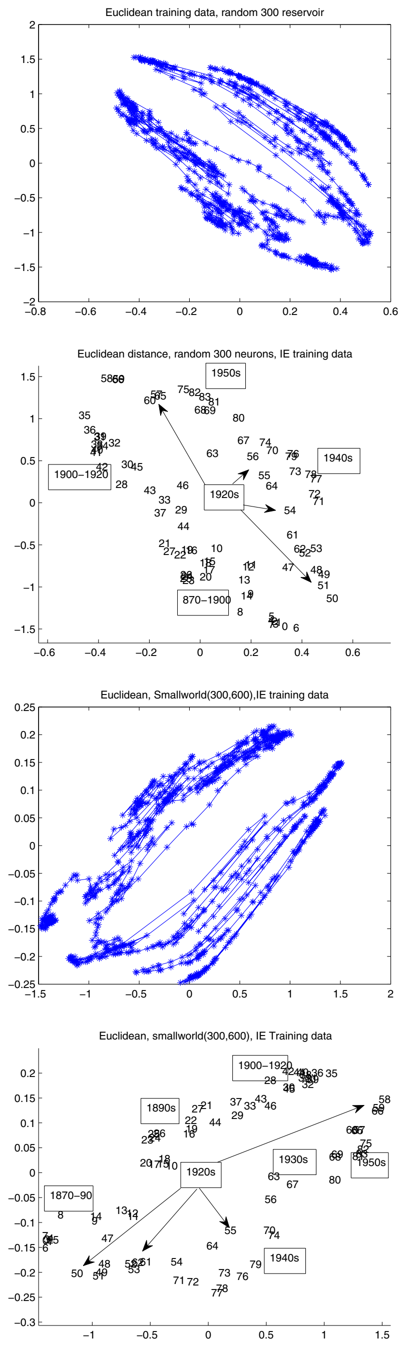

Fig. 4. Top: projections of the IE training data using a random reservoir and euclidean distance. 2nd: showing the years (1871=year 0) from this projection. 3rd: Small world reservoir and euclidean distance. Bottom: showing the years from this projection.

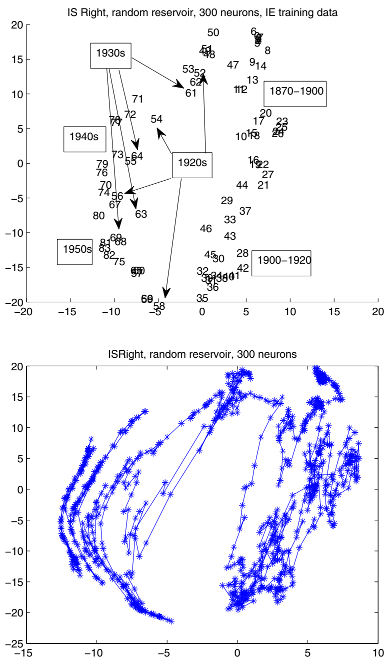

Fig. 5. Top: projection of IE training data using a random reservoir and the ISRight divergence. Bottom: showing the years from this projection.

6 and 7 which show the projections when using random reservoirs and respectively the IS Right divergence and the Bregman t=3 divergence (with $F(x) = \frac{1}{x}$). Again we can see that the structure we saw in the last section is also visible in this section showing that the method is not creating structure which does not exist in the original data. We emphasise that these are the best results achieved by an exhaustive search over our parameter space; the results from section III-B are, in contrast, easily achieved with a variety of divergences and reservoir structures.

We show in Figure 8 the projection of the training data when using the data space distances in the algorithm. This may be compared with Figure 4 in which we use the latter distances - those in the reservoir space. In Figure 4 there is somewhat more structure to be seen. The fact that there is some structure in Figure 8 does show however that we are not manufacturing relationships which do not exist in the original data.

IV. Conclusion

We have discussed visualising a specific financial time series in 3 different ways: using the neuroscale algorithm and versions of this algorithm using Bregman divergences; using a reservoir with the divergences optimising the MDS stress function 1 where the distance D_{ij} is defined either in reservoir space or in the original data space. We have shown that the last two projections are better and that the best results are achieved when D_{ij} is a function of reservoir space distances. However the fact that we could achieve similar results when D_{ij} is a function of data space distances, shows that the method is not creating structures which do not exist in the original data.

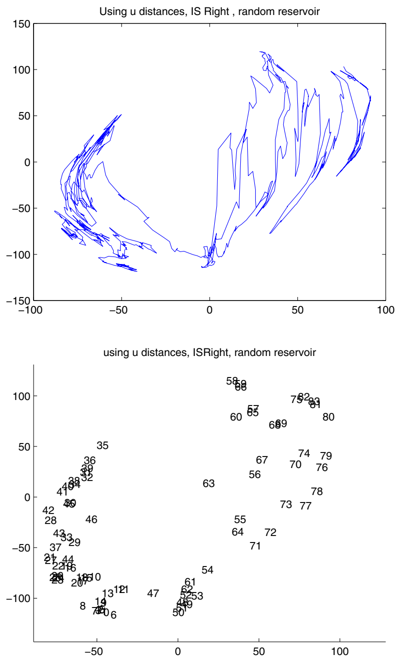

Fig. 6. Using distances in data space: IS Right

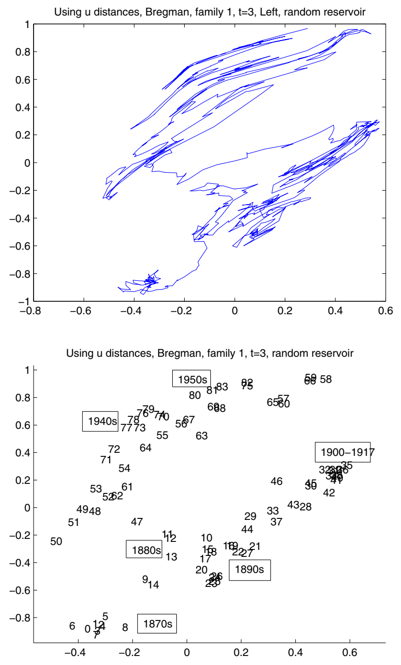

Fig. 7. Using distances in data space: t3

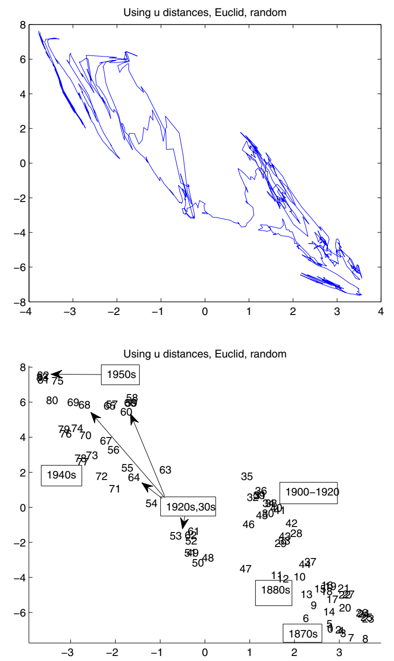

Fig. 8. Using distances in data space: the Euclidean distance.

References

[1] Herbert Jaeger. The echo state approach to analysing and training recurrent neural networks. Technical Report 148, German National Research Center for Information Technology, 2001.

[2] Benjamin Liebald. Exploration of effects of different network topologies on the esn signal crosscorrelation matrix spectrum, 2004.

[3] Mantas Lukoševičius and Hebert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3:127–149, 2009.

[4] Ali Rodan and Peter Tiňo. Minimum complexity echo state network. *IEEE Transactions on Neural Networks*, 22:131–44, 2011.

[5] R. J. Shiller. *Irrational Exuberance*. Princeton University Press, 2000,2005.

[6] J. Sun, M. Crowe, and C. Fyfe. Extending metric multidimensional scaling with bregman divergences. *Pattern Recognition*, (44):1137–1154, 2011.

[7] Jigang Sun. *Extending metric multidimensional scaling using Bregman divergences*. PhD thesis, University of the West of Scotland, 2011.

[8] Michael E. Tipping. *Topographic mappings and feed-forward neural networks*. PhD thesis, The University of Aston in Birmingham, 1996.

[9] X. Wang, M. Crowe, and C. Fyfe. Dual stream data exploration. *International Journal of Data Mining, Modelling and Management*, 2011.

[10] X. Wang, C. Fyfe, and W. Chen. Applying bregman divergences to the neuroscale algorithm. In *The United Kingdom Conference on Computational Intelligence*, 2011.

[11] Thomas Natschläger Wolfgang Maass and Henry Markram. Real-time computing without stable states: a new framework for a neural computation based on perturbations. *Neural Computation*, 14:2531–2560, november 2002.

[12] X. Wu, C. Fyfe, and T. D. Wang. The role of structure, size and sparsity in echo state networks for visualisation. In *The United Kingdom Conference on Computational Intelligence*, 2011.